

**LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK**

MODUL VIII

Exception, I/O, dan Operasi File



Oleh:

Ramdhan Wijaya

2211102208

IF-10-M

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

2024

I. DASAR TEORI

pada pembuatan program kita biasa menemukan banyak masalah, entah itu syntax error, software atau hardware error. syntax error adalah kesalahan yang dilakukan oleh programmer saat membuat program, syntax error tidak ada hubungan dengan aplikasi dan perangkat keras karena ini murni kesalahan programmer, dan untuk membenarkannya kita hanya butuh men debug lalu melakukan update terhadap program yang terjadi syntax error.

hardware error merupakan kesalahan yang terjadi Ketika, katakan keyboard kita ternyata mengalami kerusakan Ketika menjalankan program atau hanya sekedar lupa di sambungkan dengan perangkat laptop kita.

software error adalah error yang terjadi karena aplikasi dan software software pendukung lainnya bermasalah yang mengakibatkan program tidak berjalan dengan sesuaikeinginan kita. software error bukan error yang terjadi karena pogram kita berjalan tetapi program dari programmer pembuat software yang kita butuhkan untuk membuat program yang error.

error error diatas juga berhubungan dengan kesalahan intrinsic ari permodelan dunia nyata yang dilakukan oleh computer. untuk mengatasi kesalahan pembuatan dan pemotongan pada berbagai algoritma munculah disiplin ilmu Bernama Analisa numerik. jenis kesalahan ini adalah yang paling sering dilakukan oleh programmer.

exception adalah kondisi dimana Ketika user program memasukan nomor urut yang melebihi kapasitas urutan dan terjadi error namun pada exception ini kita bisa mengakali dengan prosedur try,catch dan finnaly. Semua lass exception terdapat dalam package dava.lang. semua bertipe runtime esception dan turunanya tidak harus ditangani dalam program kita. salah satu contoh error exception adalah ArrayIndexOutOfBoundsException dimana user memasukan nomor urut pada array yang dimana urutan array tidak mencapat nomor urut yang ditentukan oleh user.

penggunaan try digunakan menangani ksepso pada program kita. jadi untuk program yang sekiranya memungkinkan terjadi error kita masukan di eksepsi jika error kita akan lakukan catch jika tidak error maka badan fungsi try lah yang akan di jalankan.penggunaan throw memungkinkan untuk melempar exceptionuntuk kemudian program menyikapi exception tersebut.objek eksepsi adalah emua objek yang merupakan turunan dari class throwable.

penanganan dengan finally biasanya dimasukan ke dalam try dan catch. jadi badan fungsi yang ada pada finally akan di jalankan mau program berjalan lancar atau terjadi eksepsi.

Stream mudahnya digunakan untuk menulis/menghasilkan/outputkan tau membaca/mendapatkan suatu informasi/input. sebuah program dapat mengirim informasi dengan membuka stream ke sebuah tujuan informasi. kemudian menuliskan informasi ke tujuan secara sekuensial. pada java.io ada 2 tipe stream yaitu byte stream dan character stream.

byte stream digunakan untuk operasi I/O sedangkan character stream digunakan untuk menangani operasi I/O. jadi character stream itu teruskan oleh class reader dan writer yang merupakan abstract class dari import java.io.*;

II. GUIDED

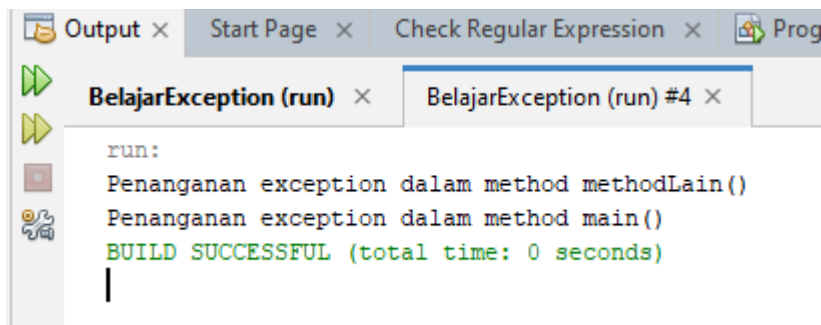
CobaThrow.java

```
/**
 *
 * @author rama
 * 2211102208
 */
public class CobaThrow {

    public static void methodLain() {
        try {
            throw new ArrayIndexOutOfBoundsException(1);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Penanganan exception dalam method
methodLain()");
            throw e;
        }
    }

    public static void main(String[] args) {
        try {
            methodLain();
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Penanganan exception dalam method
main()");
        }
    }
}
```

Output :



Keterangan :

Program di atas mendemonstrasikan penggunaan pengecualian (exception) dalam bahasa pemrograman Java, dengan fokus pada cara melempar (throw) dan menangani (catch) pengecualian di berbagai metode. Program ini terdiri dari dua metode: methodLain dan main.

Metode methodLain melempar pengecualian `ArrayIndexOutOfBoundsException` secara eksplisit menggunakan `throw new ArrayIndexOutOfBoundsException(1)`. Pengecualian ini kemudian ditangkap oleh blok `catch` dalam metode yang sama, di mana pesan "Penanganan exception dalam method methodLain()" dicetak. Setelah itu, pengecualian yang sama dilempar kembali menggunakan `throw e`, mengirimkannya ke metode pemanggil.

Di dalam metode main, methodLain dipanggil dalam blok `try`, dan pengecualian yang dilempar ulang dari methodLain ditangkap oleh blok `catch`. Blok ini kemudian mencetak pesan "Penanganan exception dalam method main()". Dengan cara ini, program menunjukkan bagaimana pengecualian yang dilempar dari satu metode dapat ditangkap dan ditangani oleh metode pemanggil, serta bagaimana pesan penanganan dapat disesuaikan di berbagai tingkatan pemanggilan metode.

Secara keseluruhan, program ini menggambarkan konsep chaining pengecualian, di mana pengecualian dapat dilewatkan dari satu metode ke metode lainnya untuk penanganan yang lebih spesifik atau lebih umum di bagian yang berbeda dari kode.

Cobafinally.java

```
/**
 *
 * @author rama
 * 2211102208
 */
public class DemoFinally {

    public static void main(String[] args) {
```

```

int x = 3;
int[] arr = {10, 11, 12};

try{
    System.out.println(arr[x]);
    System.out.println("Tidak terjadi exception");

} catch(ArrayIndexOutOfBoundsException e) {
    System.out.println("Terjadi exception");
    System.out.println(arr[x - 4]);

} finally {
    System.out.println("Program Selesai");
}
}

```

Output :

```

run:
Terjadi exception
Program Selesai
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index -1 out of bounds for length 3
    at DemoFinally.DemoFinally.main(DemoFinally.java:25)
    at C:\Users\ramad\AppData\Local\NetBeans\Cache\21\executor-snippets\run.xml:111: The following error occurred while executing this line:
    C:\Users\ramad\AppData\Local\NetBeans\Cache\21\executor-snippets\run.xml:68: Java returned: 1
BUILD FAILED (total time: 0 seconds)

```

Keterangan :

Program di atas menunjukkan cara menangani pengecualian (exception) dalam Java menggunakan blok try-catch-finally. Program ini berfokus pada bagaimana blok finally dieksekusi terlepas dari apakah pengecualian terjadi atau tidak.

Dalam metode main, sebuah variabel x diinisialisasi dengan nilai 3, dan sebuah array arr berisi tiga elemen (10, 11, 12). Blok try mencoba mengakses elemen array pada indeks x, yang dalam kasus ini adalah arr[3]. Namun, karena array arr hanya memiliki indeks dari 0 hingga 2,

mencoba mengakses arr[3] akan memicu pengecualian `ArrayIndexOutOfBoundsException`.

Ketika pengecualian ini terjadi, eksekusi dialihkan ke blok catch, yang menangani pengecualian dengan mencetak pesan "Terjadi exception". Dalam blok catch, ada upaya untuk mengakses arr[x - 4], yang berarti arr[-1], namun karena indeks negatif juga tidak valid dalam konteks array Java, ini biasanya akan memicu pengecualian lain. Namun, dalam program ini, penanganan lebih lanjut dari pengecualian tersebut tidak ditunjukkan.

Setelah blok try atau catch selesai dieksekusi, blok finally akan selalu dijalankan, mencetak pesan "Program Selesai". Hal ini memastikan bahwa pesan ini selalu ditampilkan, baik terjadi pengecualian maupun tidak.

Secara keseluruhan, program ini mengilustrasikan penggunaan blok finally untuk mengeksekusi kode yang harus dijalankan terlepas dari apakah pengecualian terjadi atau tidak, serta memberikan contoh penanganan dasar pengecualian dengan blok try-catch.

DemoStream1.java

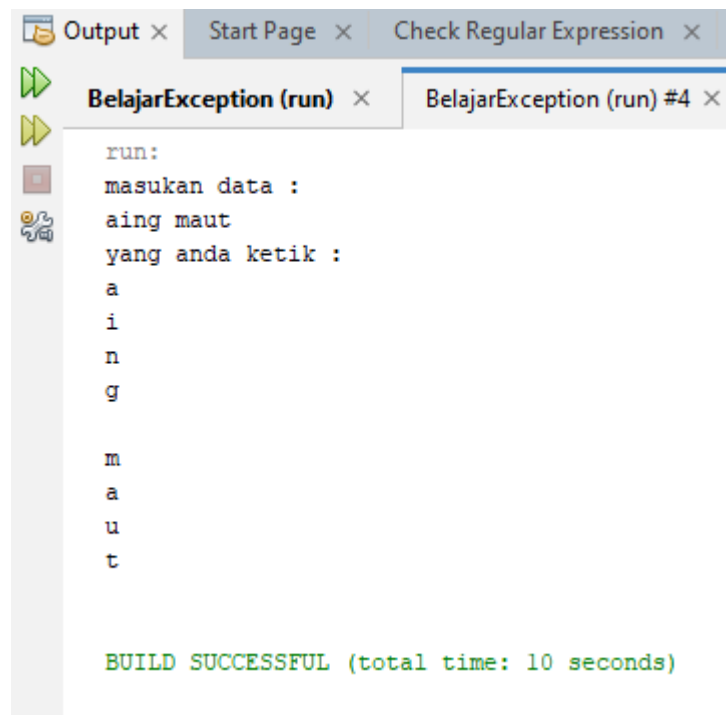
```
/**
 *
 * @author rama
 * 2211102208
 */
public class DemoStream1 {
    public static void main(String[] args){
        byte[] data = new byte[10];
        System.out.println("masukan data : ");

        try{
            System.in.read(data);

        }catch(IOException e){
            System.out.println("terjadi exception");
        }System.out.println("yang anda ketik : ");
        for(int i = 0; i<data.length;i++){
```

```
        System.out.println((char)data[i]);  
        //fungsi "(char)" adalah untuk mengubah print data menjadi tipe  
data character  
    }  
}  
}
```

Output :



```
run:  
masukan data :  
aing maut  
yang anda ketik :  
a  
i  
n  
g  
  
m  
a  
u  
t  
  
BUILD SUCCESSFUL (total time: 10 seconds)
```

Keterangan :

Program di atas adalah contoh sederhana yang mendemonstrasikan penggunaan aliran (stream) input dalam bahasa pemrograman Java, khususnya untuk membaca data dari input pengguna melalui konsol. Program ini menggunakan `System.in.read` untuk membaca input byte dari pengguna ke dalam sebuah array byte, dan kemudian menampilkan data yang dimasukkan sebagai karakter.

DemoStream2.java

```
/**
```



```

*
* @author rama
* 2211102208
*/
public class DemoStream2 {
    public static void main(String[] args ){
        byte[] data = new byte[10];
        int panjang = 0;
        System.out.println("masukan data : ");

        try{
            panjang = System.in.read(data);
            //Sistem.in.read mengembalikan panjang karakter yang
            //diinputkan oleh user termasuk enter yang dianggap 2 character

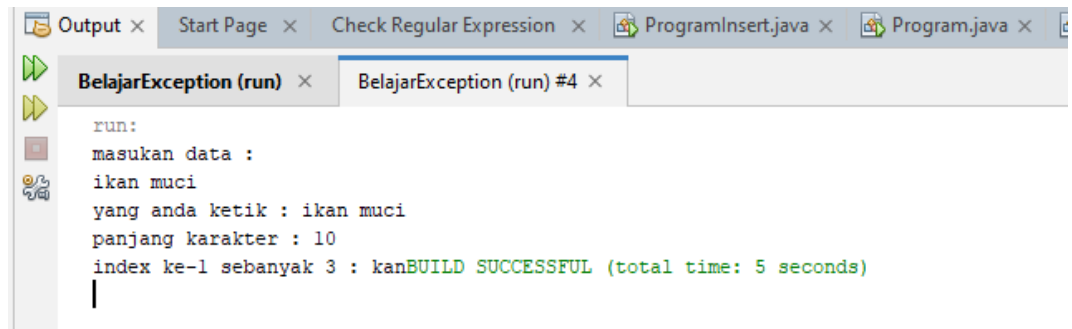
            System.out.print("yang anda ketik : ");
            System.out.write(data);
            System.out.println("panjang karakter : "+panjang);
            System.out.print("index ke-1 sebanyak 3 : ");
            System.out.write(data,1,3);//untuk membaca data lalu dari indeks
            1 sampai 3 akan di print

            }catch(IOException e){
                System.out.print("terjadi exception");
            }
        }
    }

    //write mencetak apapun tipe data yang ada
    //sedangkan print dan println mencetak data ke dalam tipe string
    //perbedaan print dan println adalah jika print tidak memberikan baris
    baru, sedangkan println akan mencetak data di baris baru

```

Output :



```
run:
masukan data :
ikan muci
yang anda ketik : ikan muci
panjang karakter : 10
index ke-1 sebanyak 3 : kanBUILD SUCCESSFUL (total time: 5 seconds)
```

Keterangan :

Program di atas adalah contoh penggunaan aliran input dan output dalam Java untuk membaca data dari pengguna dan kemudian menampilkan data tersebut. Program ini menunjukkan cara kerja metode `System.in.read` untuk input dan `System.out.write` untuk output, serta bagaimana menangani input/output dalam aplikasi Java. Deklarasi dan Inisialisasi: Program dimulai dengan mendeklarasikan sebuah array byte data berukuran 10 dan sebuah variabel integer panjang untuk menyimpan panjang data yang dibaca. Permintaan Input Pengguna: Program meminta pengguna untuk memasukkan data dengan mencetak pesan "masukan data : ". Membaca Input: Dalam blok try, metode `System.in.read(data)` digunakan untuk membaca hingga 10 byte input dari pengguna dan menyimpannya dalam array data. Metode ini juga mengembalikan jumlah byte yang dibaca, termasuk karakter "enter" yang dihitung sebagai dua karakter, yang kemudian disimpan dalam variabel panjang. Menampilkan Data yang Dimasukkan: Setelah membaca input, program mencetak "yang anda ketik : " dan menggunakan `System.out.write(data)` untuk menulis semua byte dalam array data ke output, menampilkan data yang dimasukkan oleh pengguna dalam bentuk byte. Menampilkan Panjang Data: Program kemudian mencetak panjang karakter yang dibaca dengan pesan "panjang karakter : " diikuti dengan nilai dari variabel panjang. Menampilkan Subset Data: Program menampilkan sebagian data dari indeks 1 hingga 3 dengan mencetak "index ke-1 sebanyak 3 : " dan menggunakan `System.out.write(data, 1, 3)` untuk menulis tiga byte dari array data mulai dari indeks 1. Penanganan Pengecualian: Jika terjadi kesalahan input/output, seperti kegagalan membaca data, blok catch menangkap `IOException` dan mencetak pesan "terjadi exception".

DemoStream3.java

```
/**
 *
```

```

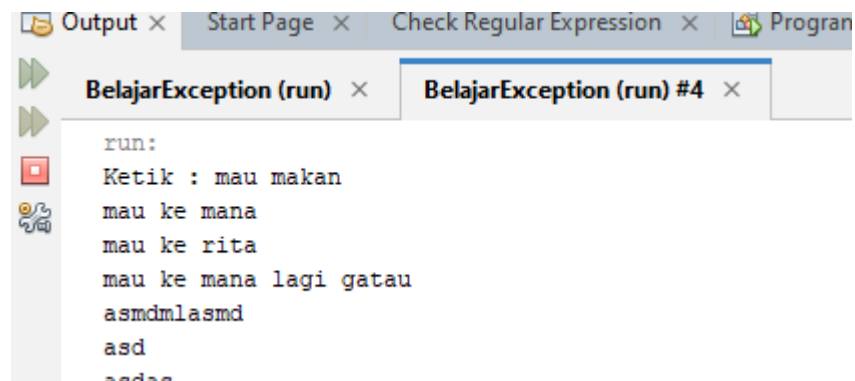
* @author rama
* 2211102208
*/
import java.io.*;

public class DemoStream3 {

    public static void main(String[] args) throws IOException {
        char data;
        String str = "";
        BufferedReader buff = new BufferedReader(new
        InputStreamReader(System.in));
        System.out.print("Ketik : ");
        data = (char) buff.read();
        while (data != '\r') {
            str += data;
            data = (char) buff.read();
        }
        System.out.println("Yang diketik : " + str);
        System.out.println("Program Selesai");
    }
}

```

Output :



```

run:
Ketik : mau makan
mau ke mana
mau ke rita
mau ke mana lagi gatau
asmdmlasmd
asd
asdas

```

Keterangan :

Program di atas adalah contoh penggunaan kelas `BufferedReader` dalam Java untuk membaca input karakter demi karakter dari pengguna melalui konsol. Program ini mengilustrasikan cara membaca input secara efisien dan membangun sebuah string dari karakter yang dibaca, hingga pengguna menekan tombol "Enter". Loop Pembacaan Input: Program menggunakan loop `while` untuk terus membaca karakter dari input hingga karakter carriage return (`\r`) terdeteksi. Setiap karakter yang dibaca ditambahkan ke

string str menggunakan operasi penggabungan string (+=). Pengguna biasanya menekan tombol "Enter" untuk mengakhiri input, yang menghasilkan karakter carriage return ('\r') dalam sistem tertentu (terutama di sistem operasi Windows).

DemoStream4.java

```
import java.io.*;

/**
 *
 * @author rama
 * 2211102208
 */
public class DemoStream4 {
    public static void main(String[] args){
        byte data;
        String namaFile = "test.txt";
        FileOutputStream fout = null;
        try{
            fout = new FileOutputStream(namaFile, true);
            //true artinya menambahkan ke dalam file, tidak menimpa file
            System.out.print("ketik : ");
            data = (byte)System.in.read();
            while (data != (byte)\r){
                fout.write(data);
                data = (byte)System.in.read();
            }

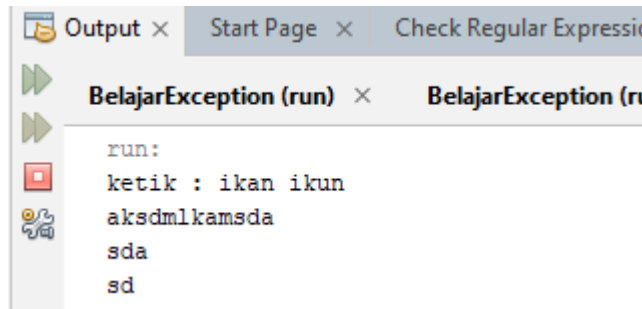
        }catch(FileNotFoundException e){
            System.out.println("file "+namaFile+" tidak dapat di create");
        }catch(IOException e){
            System.out.println("terjadi exception");
        }finally{
            if(fout != null){
                try{
                    fout.close();
                }catch(IOException e){
                    System.out.println("terjadi exception");
                }
            }
        }
    }
}
```

```

    }
}
}

```

Output :



Keterangan :

Program di atas adalah contoh penggunaan aliran output byte dalam Java untuk menulis data yang dimasukkan oleh pengguna ke dalam sebuah file. Program ini menggunakan kelas `FileOutputStream` untuk menulis byte ke dalam file, dengan fitur penambahan (append) ke file yang sudah ada, bukan menimpa (overwrite).
Membaca dan Menulis Input: Program membaca byte pertama dari input pengguna menggunakan `System.in.read()` dan menyimpannya dalam variabel `data`. Loop `while` digunakan untuk terus membaca byte dari input hingga karakter carriage return (`'\r'`) terdeteksi (biasanya dihasilkan ketika pengguna menekan tombol "Enter"). Setiap byte yang dibaca ditulis ke file menggunakan metode `fout.write(data)`.
Penanganan Pengecualian:

`FileNotFoundException`, Jika file "test.txt" tidak dapat dibuat atau diakses, blok `catch` akan menangkap pengecualian ini dan mencetak pesan "file [namaFile] tidak dapat di create".
`IOException`, Jika terjadi kesalahan input/output lainnya, blok `catch` yang kedua akan menangkap pengecualian ini dan mencetak pesan "terjadi exception".

BarangEx.java

```
/**
 *
 * @author rama
 * 2211102208
 */
import java.io.*;
public class BarangEx implements Externalizable {

    private String nama;
    private int jumlah;

    public BarangEx() { //konstruktor 1
    }

    public BarangEx(String nm, int jml) { //konstruktor2
        nama = nm;
        jumlah = jml;
    }

    public void writeExternal(ObjectOutput out) throws IOException {
        out.writeObject(nama); //string adalah tipe data referensi
        out.writeInt(jumlah);
    }

    public void readExternal(ObjectInput in) throws IOException,
        ClassNotFoundException {
        this.nama = (String) in.readObject();
        this.jumlah = in.readInt();
    }

    public String toString() {
        return "data barang: " + nama + "\n" + "jumlah barang: " + jumlah;
    }

    public static void simpanObjek(BarangEx brg) throws IOException {
        FileOutputStream fos = new FileOutputStream("dtEx.txt");
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(brg);
        oos.flush();
    }

    public static BarangEx bacaObjek() throws
        ClassNotFoundException, IOException {
        FileInputStream fis = new FileInputStream("dtEx.txt");
```

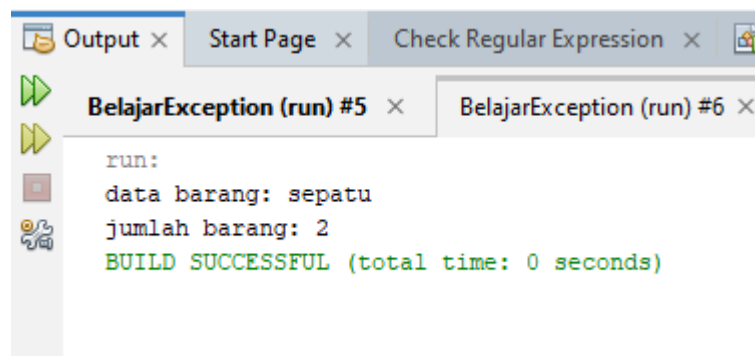
```

        ObjectInputStream ois = new ObjectInputStream(fis);
        return (BarangEx) ois.readObject();
    }

    public static void main(String[] args) throws
        ClassNotFoundException, IOException {
        BarangEx awal = new BarangEx("sepatu", 2);
        simpanObjek(awal);
        System.out.println(bacaObjek());
    }
}

```

Output :



Keterangan :

Program di atas mendemonstrasikan penggunaan serialisasi dan deserialisasi dalam Java menggunakan antarmuka Externalizable. Antarmuka ini memberikan kontrol penuh atas proses serialisasi dan deserialisasi, memungkinkan pengguna untuk menentukan secara eksplisit bagaimana objek harus disimpan dan dipulihkan. Program ini menyimpan dan membaca objek BarangEx ke dan dari file. Program ini bermanfaat untuk situasi di mana pengguna membutuhkan kontrol lebih besar atas format dan proses penyimpanan objek, serta bagaimana objek dipulihkan dari penyimpanan.

BarangSer.java

```

package DemoStream5;

/**
 *

```

```

* @author rama
* 2211102208
*/
import java.io.*;

public class BarangSer implements Serializable {

    private String nama;
    private int jumlah;

    public BarangSer(String nm, int jml) {
        nama = nm;
        jumlah = jml;
    }

    public void tampil() {
        System.out.println("nama barang: " + nama);
        System.out.println("jumlah barang: " + jumlah);
    }

    public void simpanObject(BarangSer ob) {
        try {
            FileOutputStream fos = new FileOutputStream("dtBrg.txt");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(ob);
            oos.flush();
        } catch (IOException ioe) {
            System.err.println("error" + ioe);
        }
    }

    public void bacaObject(BarangSer obb) {
        try {
            FileInputStream fis = new FileInputStream("dtBrg.txt");
            ObjectInputStream ois = new ObjectInputStream(fis);
            while ((obb = (BarangSer) ois.readObject()) != null) {
                obb.tampil();
            }
        } catch (IOException ioe) {
            System.exit(1);
        } catch (Exception e) {
            System.exit(1);
        }
    }

    public static void main(String[] args) {

```




```

        BarangSer a1 = new BarangSer("Baju", 5);
        a1.simpanObject(a1);
        a1.bacaObject(a1);
    }
}

```

Output :



```

Output x Start Page x Check Regular Expression x ProgramInsert.java x Program.java x ProgramResultSetStatement.java x BarangPreparedStatementSen
BelajarException (run) #5 x BelajarException (run) #6 x
run:
nama barang: Baju
jumlah barang: 5
E:\Users\ramad\AppData\Local\NetBeans\Cache\21\executor-snippets\run.xml:111: The following error occurred while executing this line:
C:\Users\ramad\AppData\Local\NetBeans\Cache\21\executor-snippets\run.xml:68: Java returned: 1
BUILD FAILED (total time: 0 seconds)

```

Keterangan :

Program di atas adalah contoh penggunaan serialisasi dan deserialisasi di Java menggunakan antarmuka Serializable. Kelas BarangSer merepresentasikan sebuah objek barang dengan nama dan jumlah, dan menyediakan metode untuk menyimpan dan membaca objek dari sebuah file.

TryCatch1.java

```

/**
 *2211102208
 * @author rama
 */
public class TryCatch1 {
    public static void main(String[] args) {
        try{
            int[] arr = new int [1];
            System.out.println(arr[1]);
            System.out.println("Baris ini tidak akan dieksekusi, karena
statement baris diatas terjadi exception");

        }catch(Exception ex){
            System.out.println("Terjadi exception karena indeks di luar
kapasitas array");
        }
    }
}

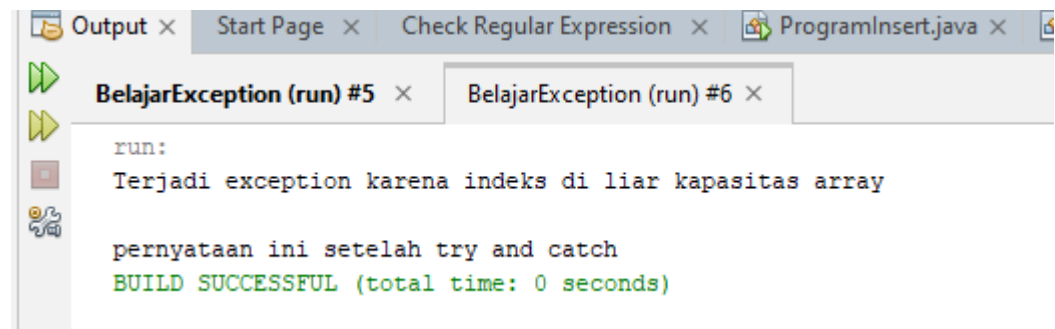
```

```

        System.out.println("\npernyataan ini setelah try and catch ");
    }
}

```

Output :



Keterangan :

Program di atas adalah contoh sederhana penggunaan blok try-catch dalam Java untuk menangani pengecualian (exception) yang mungkin terjadi selama eksekusi program. Program ini mencoba mengakses elemen array di luar batasannya, yang menyebabkan pengecualian `ArrayIndexOutOfBoundsException`. Blok Try:

```
try { ... }
```

Bagian ini mencoba untuk menjalankan kode di dalamnya dan jika terjadi pengecualian, akan dilempar ke blok catch.

```
System.out.println(arr[1]);
```

Kode ini mencoba mengakses elemen kedua dari array (`arr[1]`), yang tidak ada karena array hanya memiliki satu elemen (`arr[0]`). Ini akan menyebabkan pengecualian `ArrayIndexOutOfBoundsException`.

TryCatch2.java

```

/**
 *
 * @author rama
 */
public class TryCatch2 {

```

```

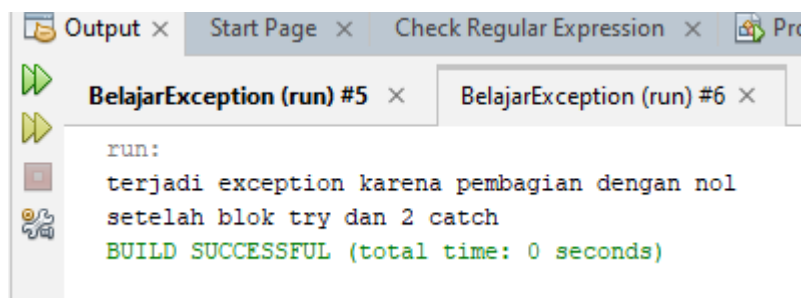
public static void main (String[] args){
    try{
        int x = args.length; //merupakan banyak argument
        int y = 100/x;
        int[] arr = {10,11};
        y = arr[x];
        System.out.println("tidak terjadi exception");
    }catch(ArithmeticException ex){
        System.out.println("terjadi exception karena pembagian dengan nol");
    }

    }catch(ArrayIndexOutOfBoundsException e){
        System.out.println("terjadi exception karena indeks di luar kapasitas array");
    }

    }
    System.out.println("setelah blok try dan 2 catch");
}
}

```

Output :



```

Output x Start Page x Check Regular Expression x Pr
BelajarException (run) #5 x BelajarException (run) #6 x
run:
terjadi exception karena pembagian dengan nol
setelah blok try dan 2 catch
BUILD SUCCESSFUL (total time: 0 seconds)

```

Keterangan :

Program ini menunjukkan bagaimana menggunakan blok try-catch dengan beberapa blok catch untuk menangani berbagai jenis pengecualian yang mungkin terjadi selama eksekusi program. Dengan menangkap pengecualian, program dapat memberikan pesan yang jelas tentang kesalahan yang terjadi dan melanjutkan eksekusi kode berikutnya dengan lancar. Ini adalah praktik yang baik untuk meningkatkan keandalan dan keselamatan program.